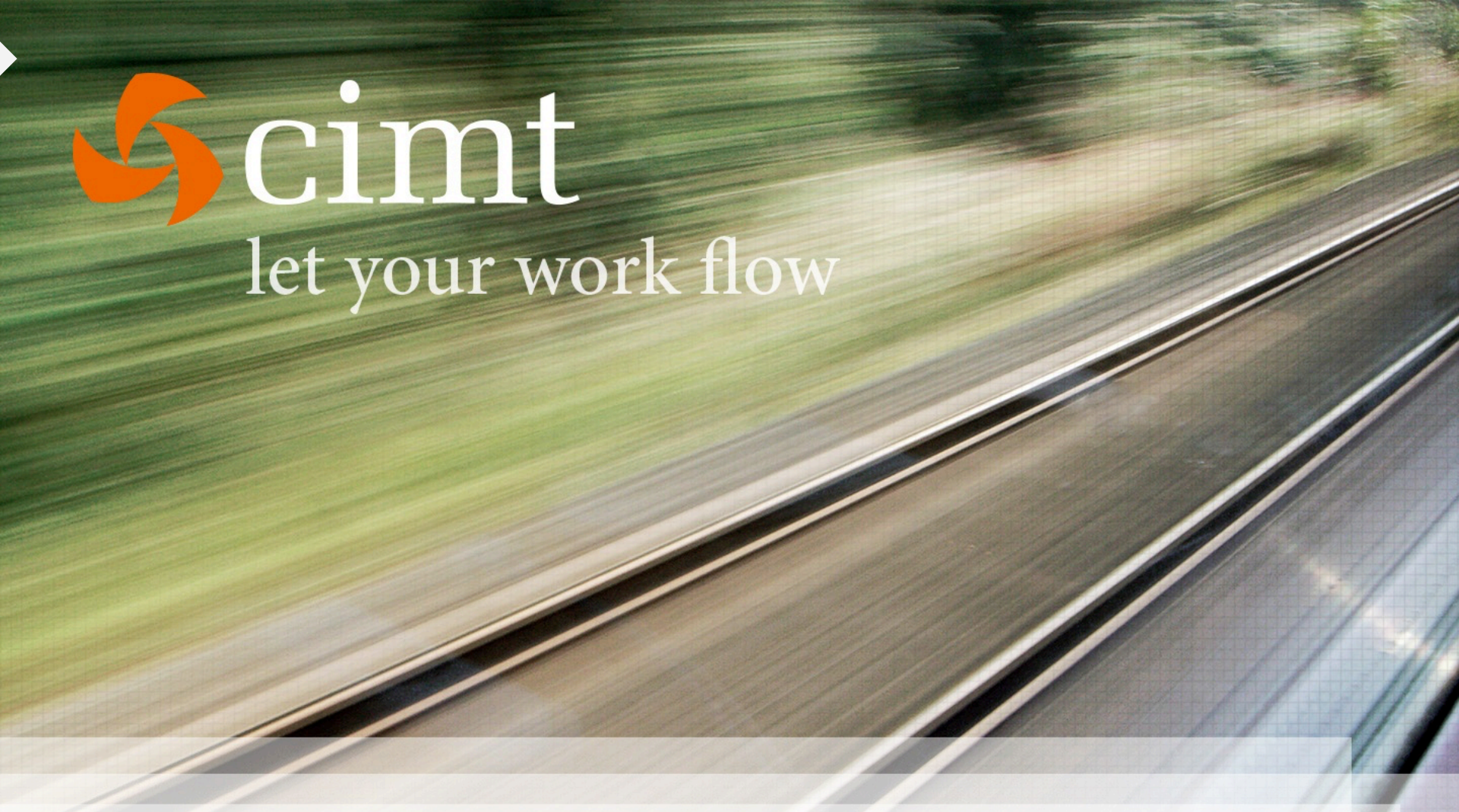



# COMPETENCE FOR BI, BIG DATA AND ESB PROJECTS

## How to steer and monitor jobs. Pattern and Best Practices.

# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Different Kind of Data Integration Jobs

- Simple Jobs:
  - Does not need input parameters to know which data should be processed
  - Does not start on a particular piece of data
  - Does not need to partition the data to process
  - Example: Any job
- Incremental Loads
  - Proceed data which follows the data previously proceed
  - Depends and a sequential criteria to know where to start
  - Example: Growing source tables
- Partitioned Data Loads
  - Data has a defined start and end
  - Data items to process separately
  - Data are pre-packaged / partitioned
  - Example: File processing / Calendar based data processing

# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Requirements for Data Integration Jobs

- Restart ability
  - Jobs should detect it self what is the work to do
  - Jobs should finish its work completely or nothing
  - It must be allowed to start a job without problems
- Limited Resource Allocation
  - Jobs should not need more the 1GB RAM
  - Jobs have to process huge amount of data in a streaming like way
  - Jobs should use a closed data range to process
- Projectable processing
  - It is by far more important to have a system in which the administrator can monitor the progress than a system which is some percentage faster but does not allow the recognise any progress.
  - It is also a design goal to spread the workload over a longer period of time instead of having a huge peak which makes the system a irresponsible for a unknown period of time.

# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Talend Components and Data Model

- The system consists of 4 Talend components:
  - tJobInstanceStart:
    - Creates an entry in the JOB_INSTANCE_STATUS table when the job starts.
    - Initialises the logging framework
    - Provides information about previous job runs
  - tJobInstanceEnd:
    - Collects measurements
    - Collects status information about the job run
    - Updates the entry in the JOB_INSTANCE_STATUS table
  - tJobDataRangeScanner:
    - Scans a flow for min/max values to set them as time range start/end and value range start/end.
  - tJobInstanceLifeCheck
    - Checks if some open entries in the JOB_INSTANCE_STATUS table are not alive anymore and close them up.
- All components are described in detail in the PDF documentation

# Talend Components and Data Model

» The data model consists of 4 core tables and one optional table:
  - JOB_INSTANCE_STATUS
    - Job meta data
    - Timestamps
    - Counters
    - OS information
    - Return code and messages
  - JOB_INSTANCES_LOGS
    - Log messages
  - JOB_INSTANCE_COUNTERS
    - Detail counters
  - JOB_INSTANCE_CONTEXT
    - Context variables at the start and at the end of the job run.
  - JOB_CALENDAR
    - A full fledged calendar with financial date ranges

# Architecture hints

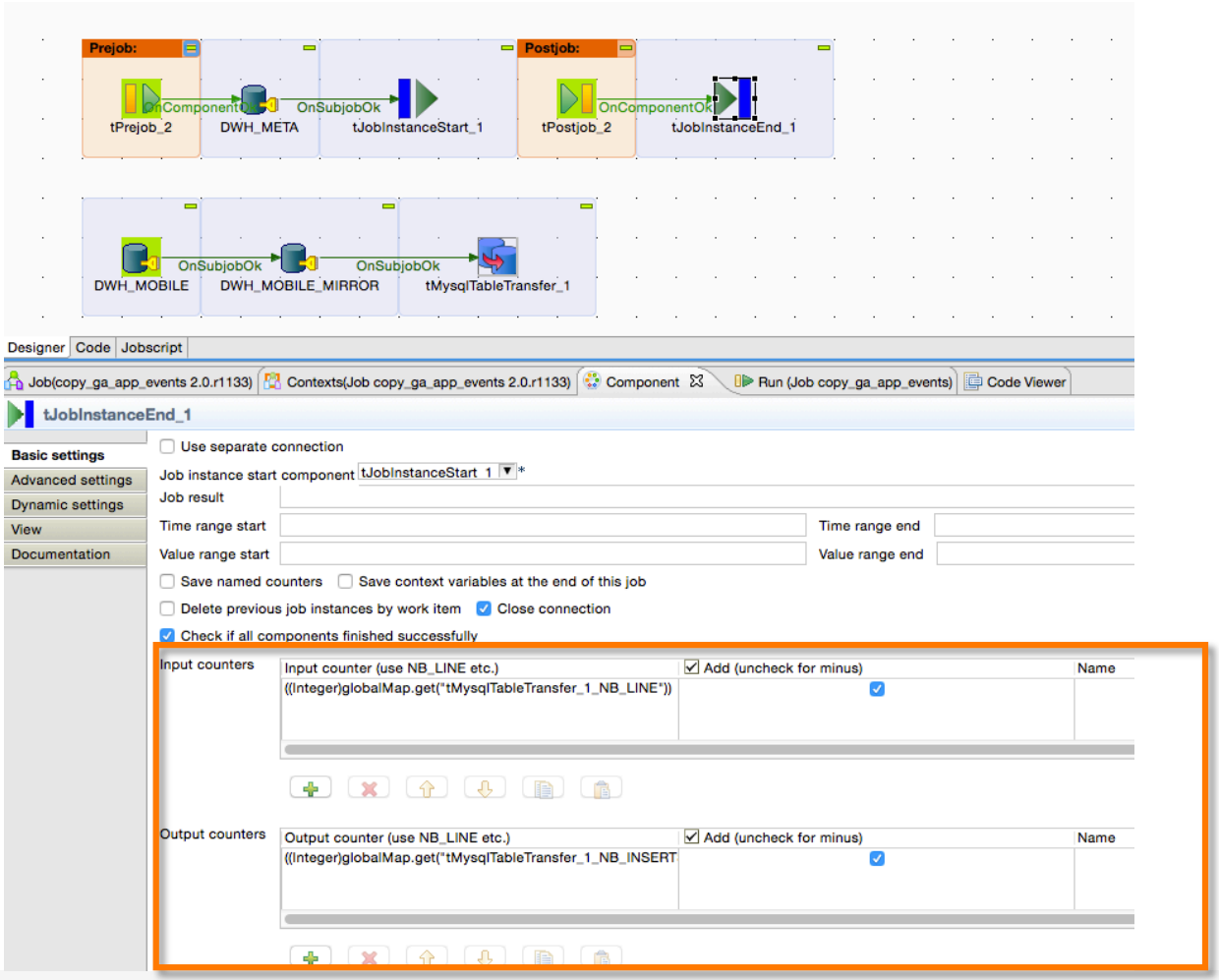
- Take care of the kind of generation of the JOB_INSTANCE_ID
- Add the column JOB_INSTANCE_ID (case sensitive is not an issue) to the actual data tables to provide data lineage.
- Sometimes it is supposed to have 2 different columns job_instance_id_insert and job_instance_id_update if you want to track which job has created and updated the data.
- Use the field option of the output components to send the job_instance_id according to the statement type.
- Decide if the table names fits to the customers naming rules. If not set alternative table names.
- Decide if the amount of job runs are very high and use more than one schema + tables for the monitoring. E.g. use a dedicated schema + tables for the staging jobs and another schema + tables for the core and mart jobs.
- Another approach is to establish different schemas for different (not interoperating) parts of the system.
- Probably it is supposed to have partitioned table for JOB_INSTANCE_STATUS (e.g. with the month as partitions or use the project as partitioning criteria)

# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Using the framework for a simple job

- It provides runtime information
- Allows to report the job runs in a unified way




Counters:
Use the return values from the components.
Hint: Use CTRL+Space

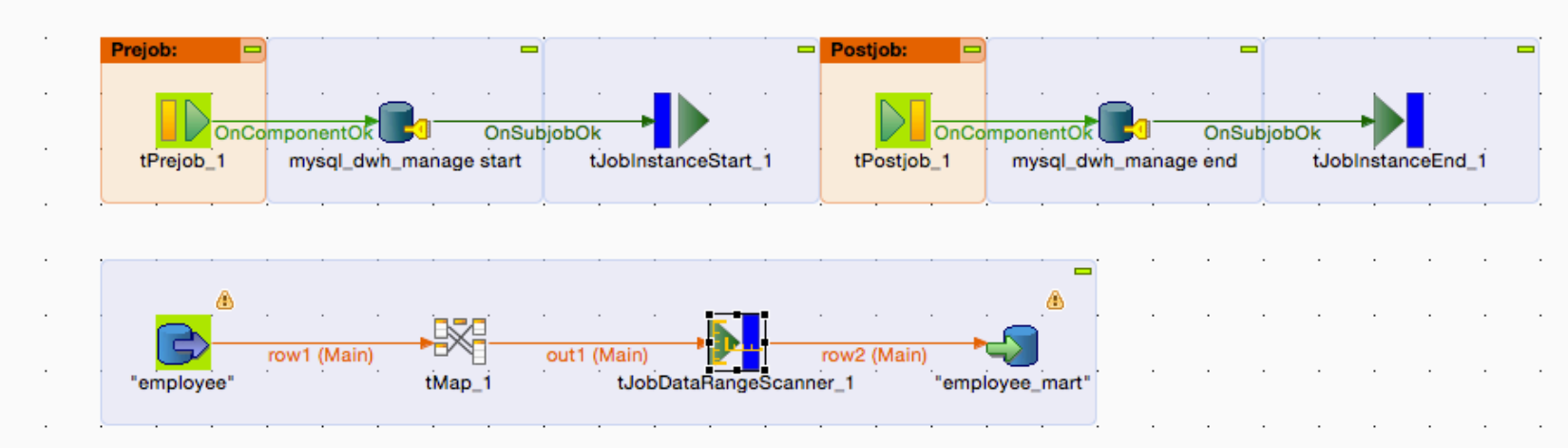
# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Using the framework for incremental steering
# Source column based

- Use the data ranges from the previous run to select the input data for the current run
- Decide which kind of value do you can use to identify the new / changed records in the source
  - Time based ranges (Date type)
  - Value based ranges (Long or String typed)
- Detect the range within the job run with tJobDataRangeScanner (this avoids to measure it with a probably expensive select in the source or target table).
- Especially for time ranges it is very important NOT to rely on the local job run time. The time on different servers potentially differs.
- We have to measure the last time stamp within the data because after the job has been finished, it is possible to get data with an timestamp BEFORE the timestamp of the job end – e.g. because of delayed mirroring mechanism.

# Using the framework for incremental steering – job design Source column based




It is important to measure the data ranges near the target to be sure these Records are persisted. A good idea is also using a transaction.

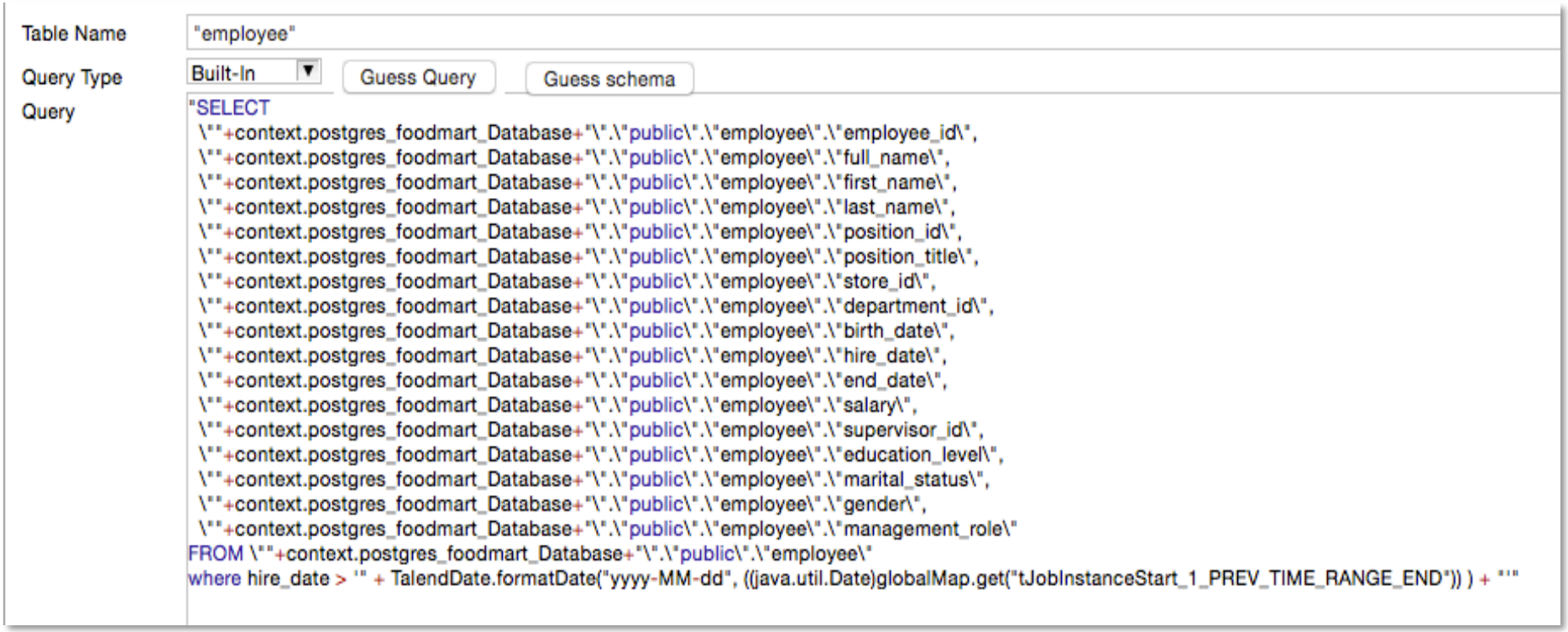
Information about previous job runs

- [x] Return last instance results
- [x] Last successful
- [x] Last must have data inserted or deleted
- [ ] For the current work item
- [ ] Collect Job Instance IDs running after previous run

| | |
|---|---|
| OK Result Codes (Comma separated list) | "0,100" |
| Replacement for prev. job-instance-id if job runs first time | 0l |
| Replacement for prev. job-start-date if job runs first time | TalendDate.parseDate("yyyy-MM-dd", "1970-01-01") |
| Replacement for prev. time-range-end if job runs first time | TalendDate.parseDate("yyyy-MM-dd", "1970-01-01") |
| Replacement for prev. value-range-end if job runs first time | "0" |
| Replacement for prev. result-item if job runs first time | "0" |

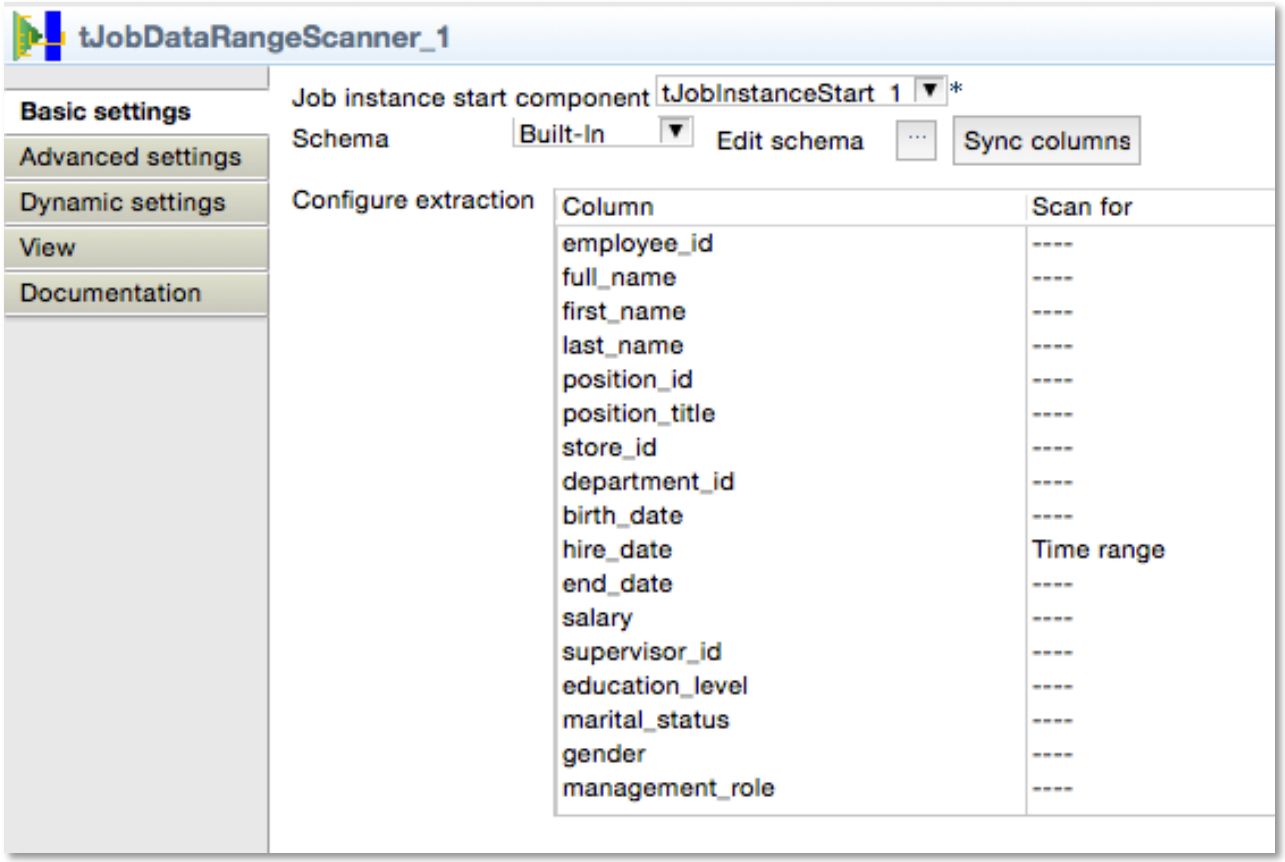
Necessary settings of the tJobInstanceStart component to get the information about the last job run.

# Using the framework for incremental steering – job design Source column based




Settings of the source database input component.
Here we use the time range end value of the previous run to select the data newer as we already got in the last run.

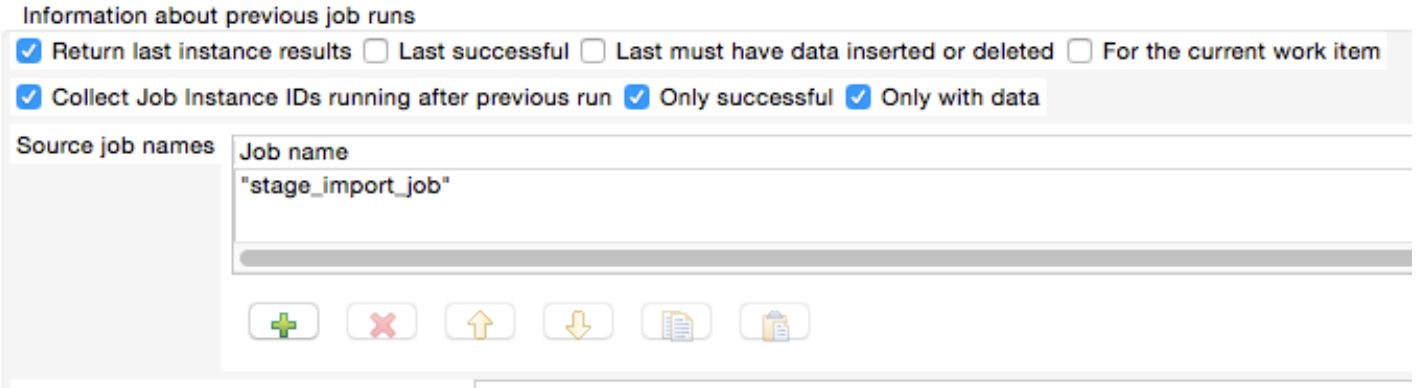
# Using the framework for incremental steering – job design Source column based




The component tJobDataRangeScanner does not have any impact on its flow. It simply scans columns on the min/max value or timestamps and sets these values as time or value range min/max values.

# Using the framework for incremental steering – job design Job_instance_id based

tJobInstanceStart settings to return the list of job instance ids of all job runs after the previous run of this job.




If the source and target table does have job_instance_id columns, these column can be used to detect changed records.

The new or changed source records can be selected with:

```
where job_instance_id in (" +
((String)globalMap.get("tJobInstanceStart_1_SOURCE_JOB_INSTANCE_ID_LIST")) + ")"
```

With this approach it is possible to build a fully generic way to detect the necessary records to process in data ware house systems.

# Agenda

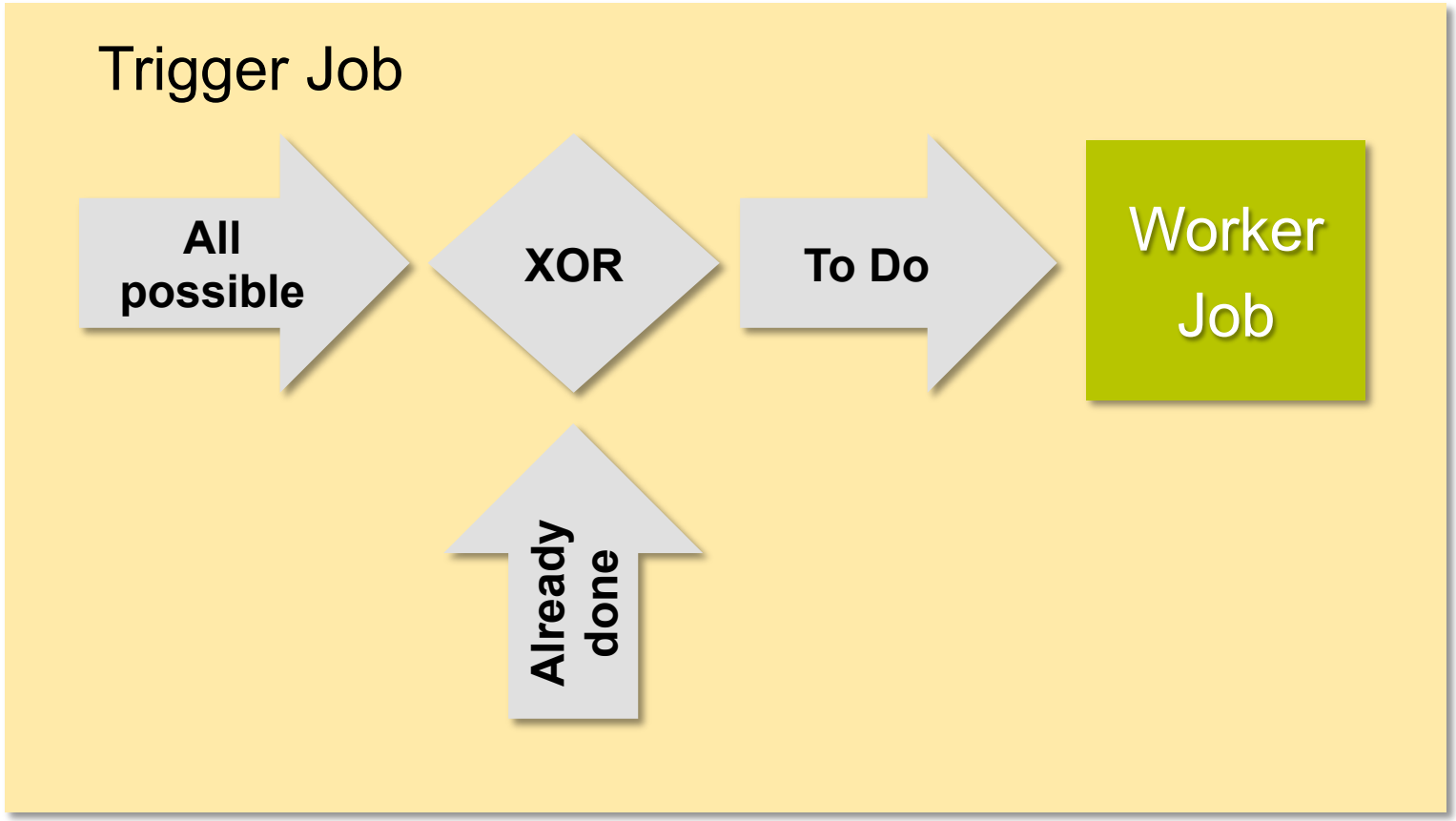
| | |
|---|---|
| 1 | Different Kind of Data Integration Jobs |
| 2 | Requirements |
| 3 | Overview about Components and Data Model |
| 4 | Using Components for Simple Jobs |
| 5 | Using Components for Incremental Loads |
| 6 | Using Components for Partinioned Data Loads |
| 7 | Monitoring Reports |
| 8 | Logging |
| 9 | Outlook |

# Using the framework for partitioned data loads

- The source data have to have an dedicated realm
- Examples:
  - Data are content of an file
  - Data are related to a date or any possible incremental value like countries, distinct ids.
- On data element (file, date etc.) is called a work item
- The target data should not rely on the previous loaded data.
- The pattern needs to know all possible work items and the work items already successful processed.
- The amount of possible work items can be become a huge number and should therefore reduced to a reasonable amount like the dates of the last year.
- This pattern consists of 2 jobs: a trigger job which selects the work items to do and a worker job which process the data of one work item.
- Pattern allows the parallel processing of work items as long as they can processed separately

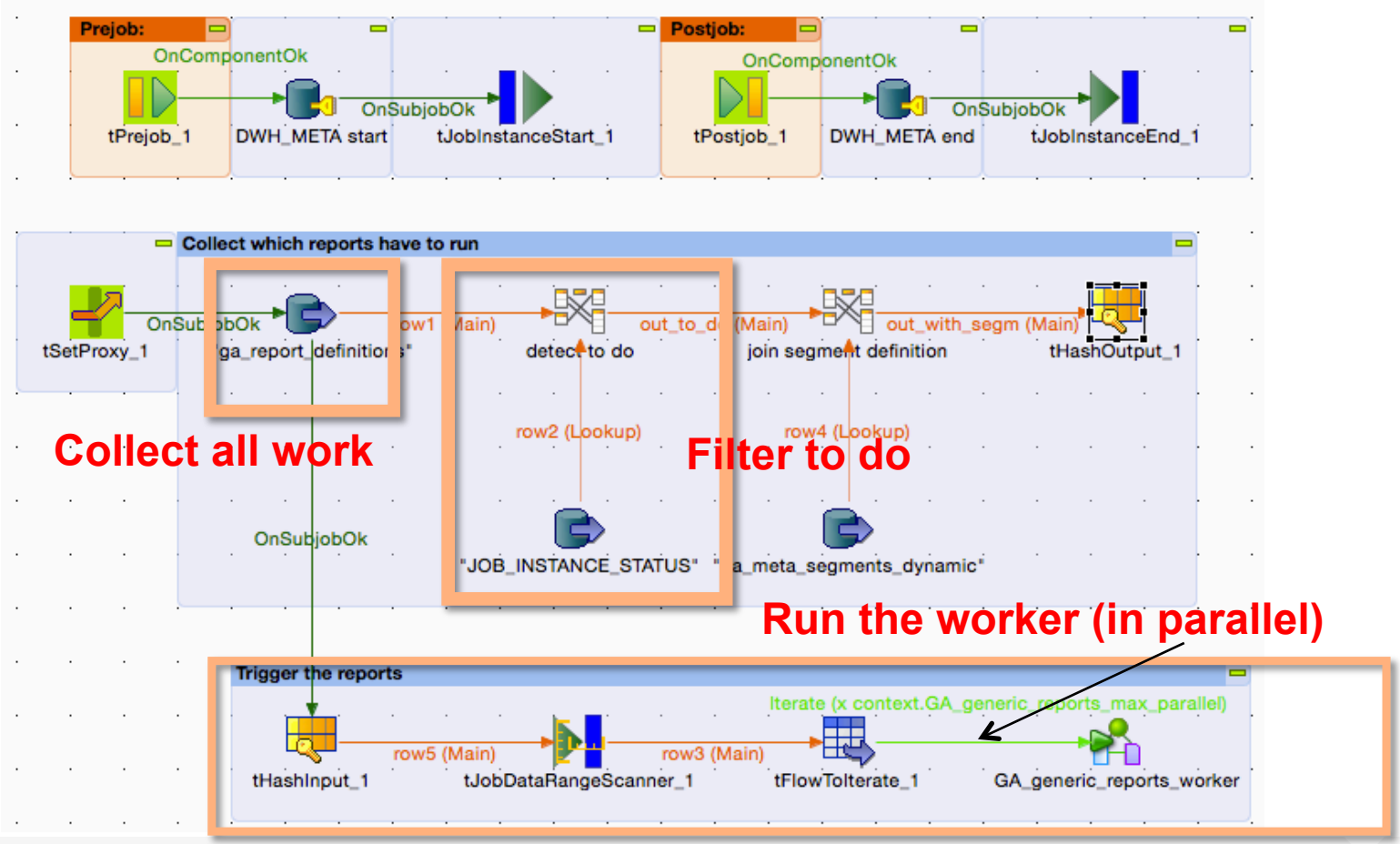
# Using the framework for partitioned data loads

» Trigger and worker job pattern.

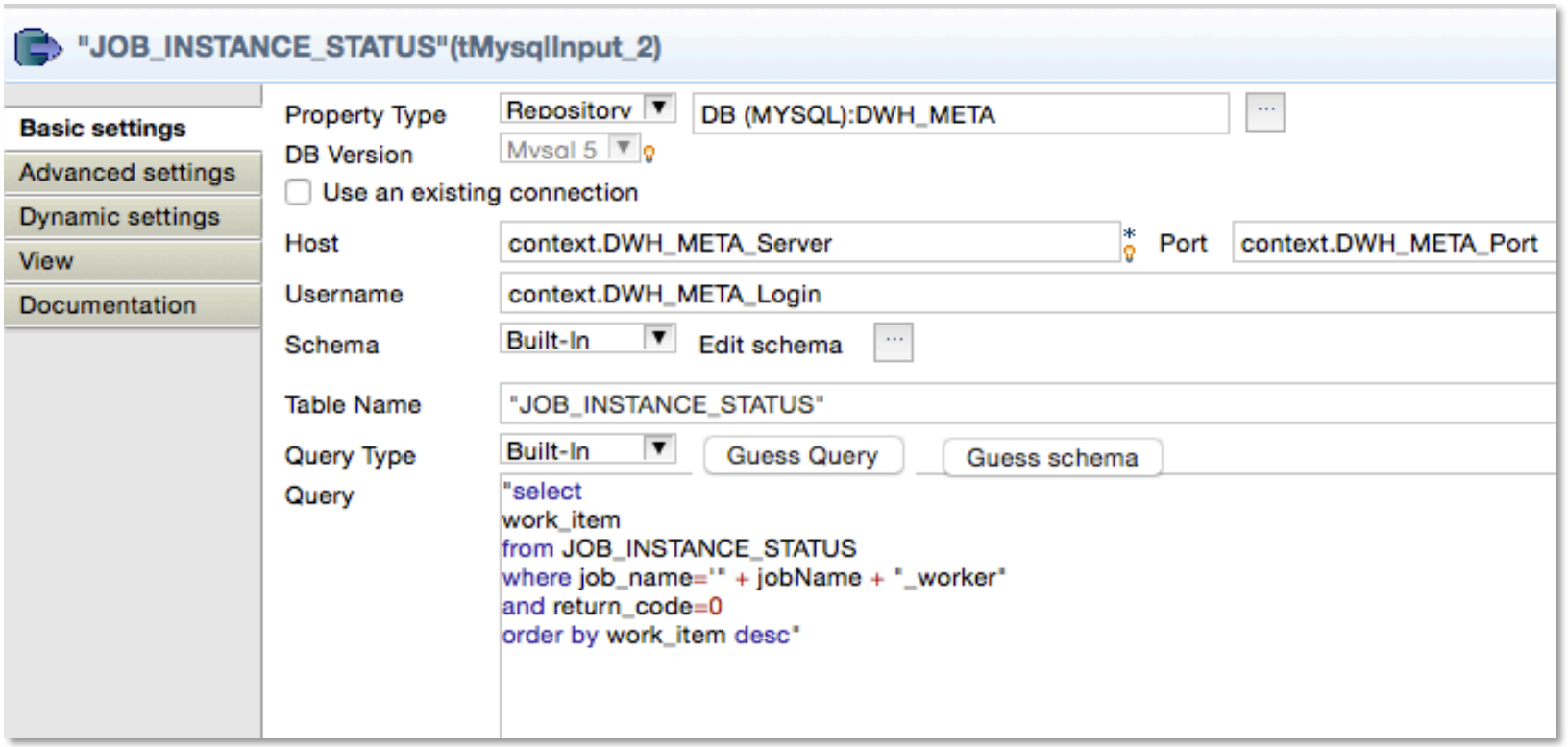


# Using the framework for partitioned data loads

- Select the total amount of work items
- Select the amount of done work items
- Build the difference and start the worker job(s - in case of parallel runs)

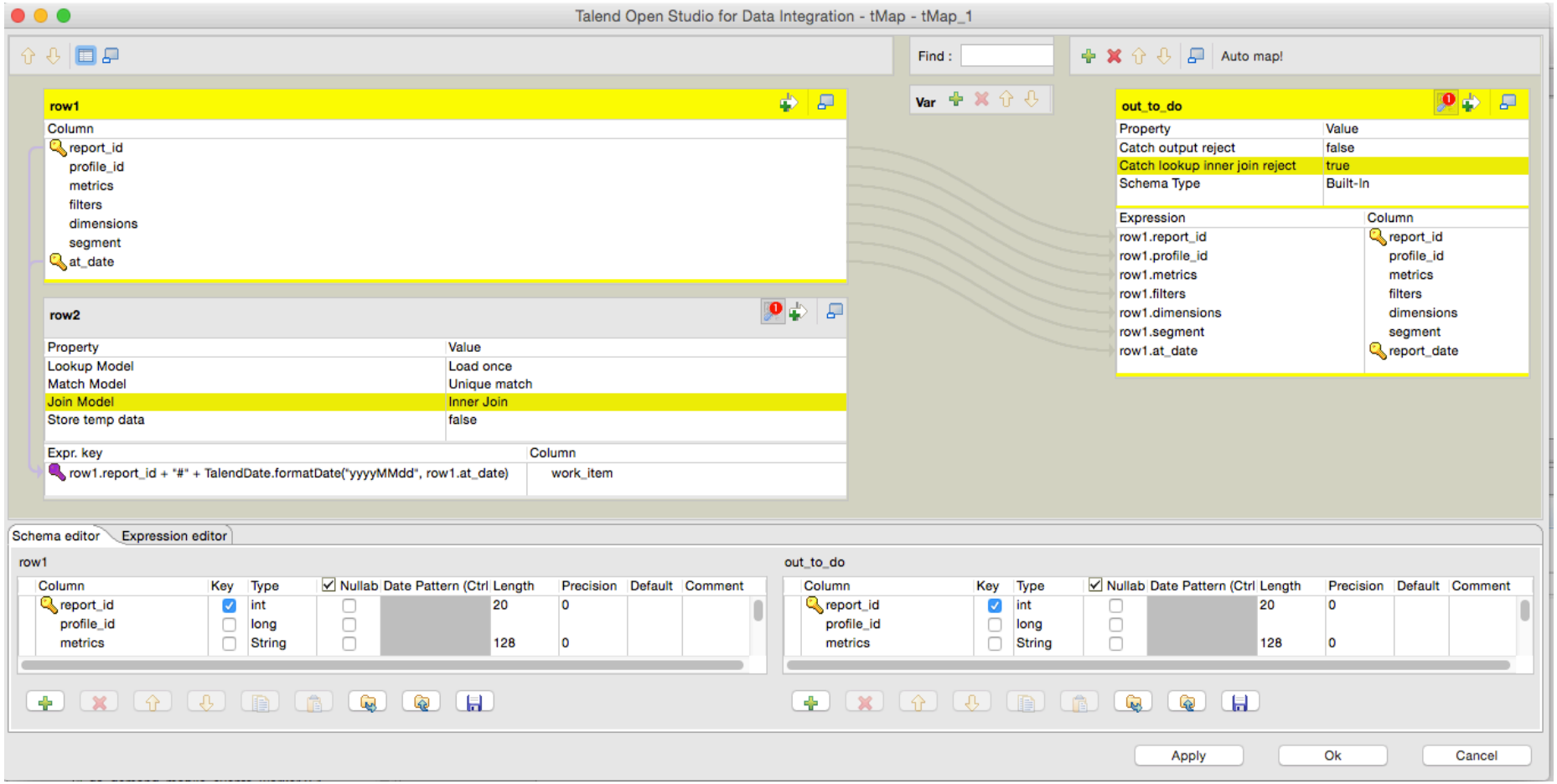


# Using the framework for partitioned data loads
# Select the done work items




If the worker job is named like trigger job + _worker
it is very easy to select the done work items.

# Using the framework for partitioned data loads
# Select the work items to do




The work items (total and done will be joined and the not matching items are the work items to do. Please notice the inner join and the inner join reject option!

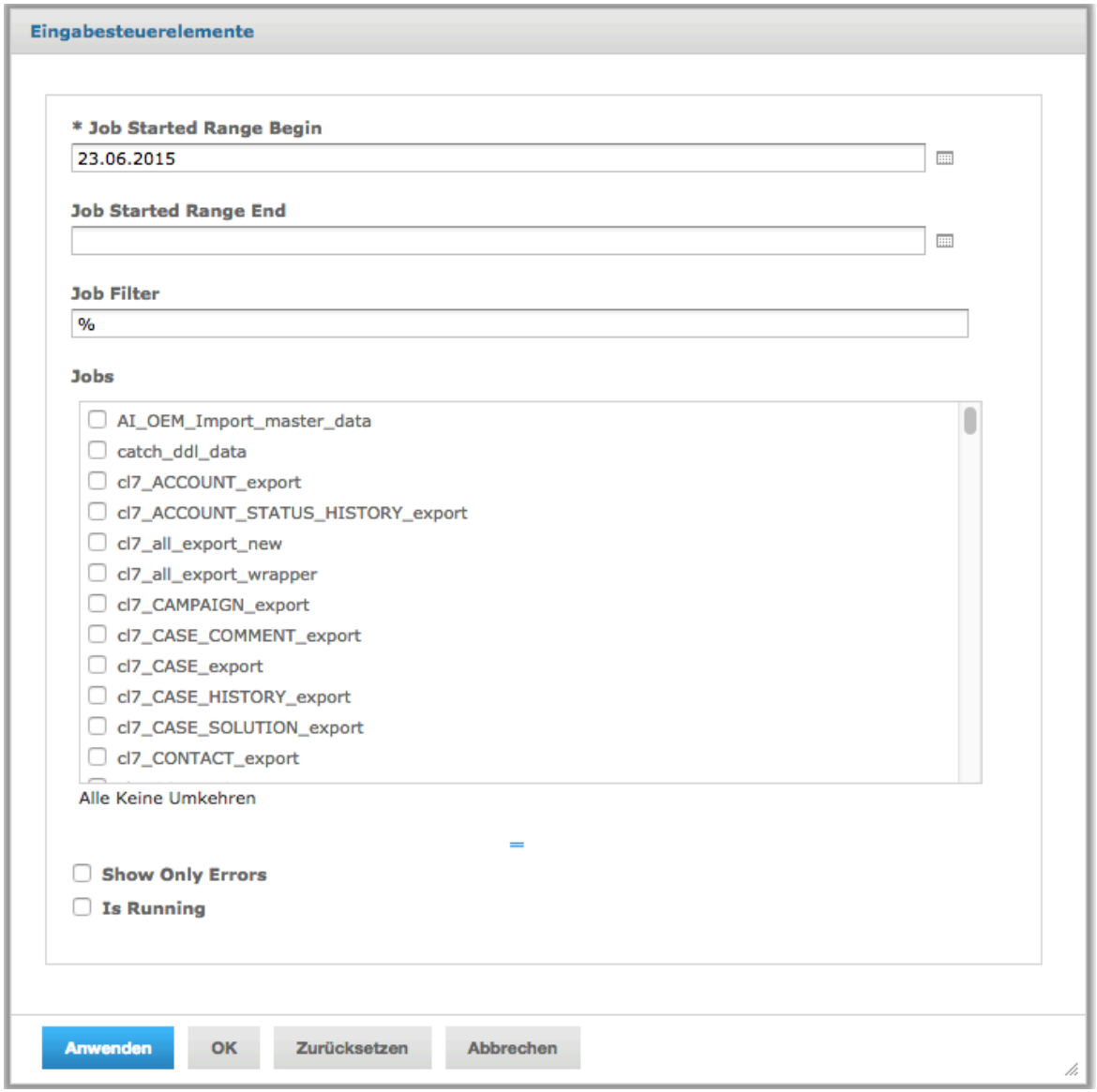
# Agenda

| | |
|---|---|
| 1 | Different Kind of Data Integration Jobs |
| 2 | Requirements |
| 3 | Overview about Components and Data Model |
| 4 | Using Components for Simple Jobs |
| 5 | Using Components for Incremental Loads |
| 6 | Using Components for Partinioned Data Loads |
| 7 | Monitoring Reports |
| 8 | Logging |
| 9 | Outlook |

# Monitoring Reports

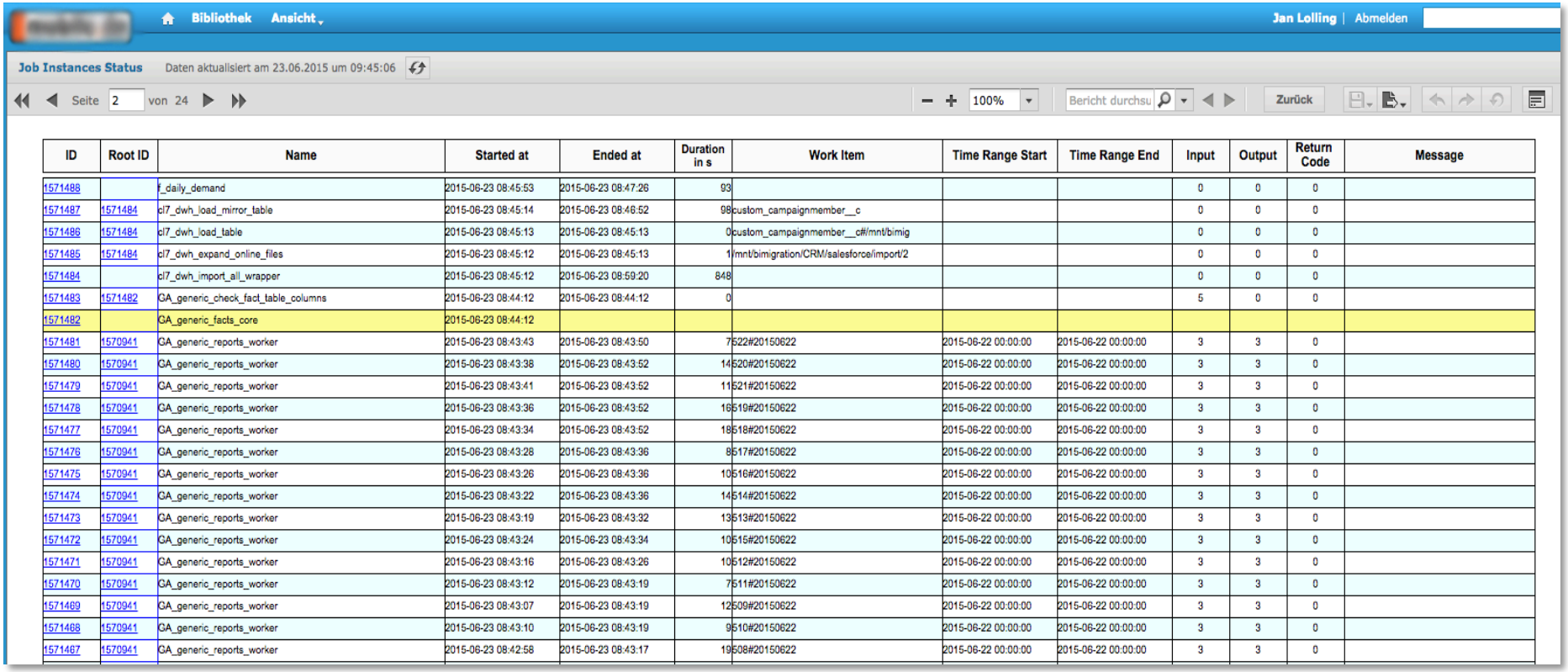
Choose the parameters:

- Start and end date
- Filter job names Select the jobs to report (nothing select means all)
- Restrict to failed jobs
- Restrict to running jobs



# Monitoring Reports – Job List

Report list of selected jobs in the selected date range with the options.

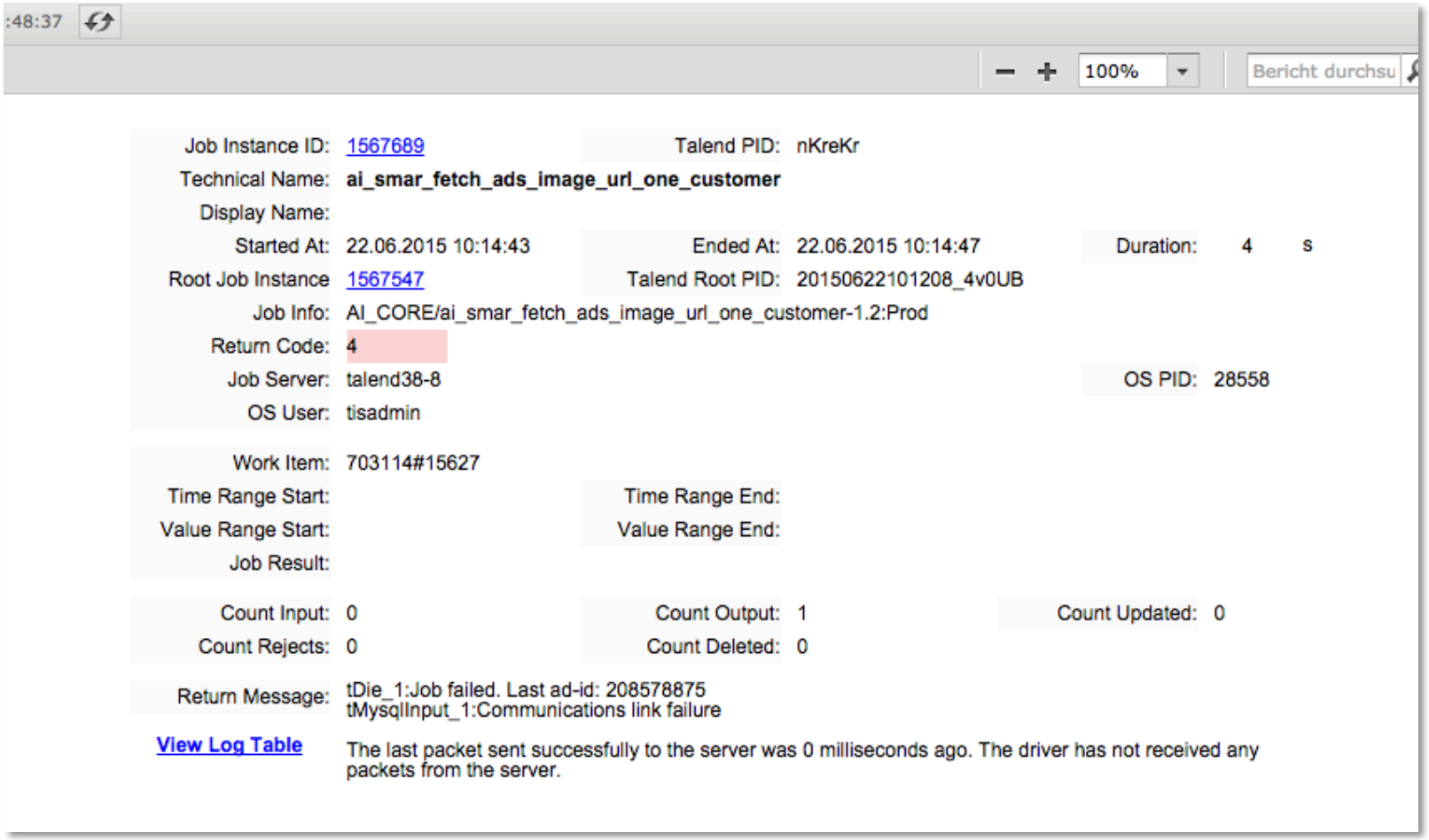



| ID | Root ID | Name | Started at | Ended at | Duration in s | Work Item | Time Range Start | Time Range End | Input | Output | Return Code | Message |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1571488 | | f_daily_demand | 2015-06-23 08:45:53 | 2015-06-23 08:47:26 | 93 | | | | 0 | 0 | 0 | |
| 1571487 | 1571484 | cl7_dwh_load_mirror_table | 2015-06-23 08:45:14 | 2015-06-23 08:46:52 | 98 | custom_campaignmember__c | | | 0 | 0 | 0 | |
| 1571486 | 1571484 | cl7_dwh_load_table | 2015-06-23 08:45:13 | 2015-06-23 08:45:13 | 0 | custom_campaignmember__c#/mnt/bimig | | | 0 | 0 | 0 | |
| 1571485 | 1571484 | cl7_dwh_expand_online_files | 2015-06-23 08:45:12 | 2015-06-23 08:45:13 | 1 | /mnt/bimigration/CRM/salesforce/import/2 | | | 0 | 0 | 0 | |
| 1571484 | | cl7_dwh_import_all_wrapper | 2015-06-23 08:45:12 | 2015-06-23 08:59:20 | 848 | | | | 0 | 0 | 0 | |
| 1571483 | 1571482 | GA_generic_check_fact_table_columns | 2015-06-23 08:44:12 | 2015-06-23 08:44:12 | 0 | | | | 5 | 0 | 0 | |
| 1571482 | | GA_generic_facts_core | 2015-06-23 08:44:12 | | | | | | | | | |
| 1571481 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:43 | 2015-06-23 08:43:50 | 7 | 522#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571480 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:38 | 2015-06-23 08:43:52 | 14 | 520#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571479 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:41 | 2015-06-23 08:43:52 | 11 | 521#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571478 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:36 | 2015-06-23 08:43:52 | 16 | 519#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571477 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:34 | 2015-06-23 08:43:52 | 18 | 518#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571476 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:28 | 2015-06-23 08:43:36 | 8 | 517#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571475 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:26 | 2015-06-23 08:43:36 | 10 | 516#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571474 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:22 | 2015-06-23 08:43:36 | 14 | 514#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571473 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:19 | 2015-06-23 08:43:32 | 13 | 513#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571472 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:24 | 2015-06-23 08:43:34 | 10 | 515#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571471 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:16 | 2015-06-23 08:43:26 | 10 | 512#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571470 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:12 | 2015-06-23 08:43:19 | 7 | 511#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571469 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:07 | 2015-06-23 08:43:19 | 12 | 509#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571468 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:43:10 | 2015-06-23 08:43:19 | 9 | 510#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |
| 1571467 | 1570941 | GA_generic_reports_worker | 2015-06-23 08:42:58 | 2015-06-23 08:43:17 | 19 | 508#20150622 | 2015-06-22 00:00:00 | 2015-06-22 00:00:00 | 3 | 3 | 0 | |

Yellow highlighted jobs are running.
Red highlighted jobs have been failed.

# Monitoring Reports – Job Details

Details are shown after follow the link on the Job Instance ID



| | | | | | |
|---|---|---|---|---|---|
| Job Instance ID: | 1567689 | Talend PID: | nKreKr | | |
| Technical Name: | **ai_smar_fetch_ads_image_url_one_customer** | | | | |
| Display Name: | | | | | |
| Started At: | 22.06.2015 10:14:43 | Ended At: | 22.06.2015 10:14:47 | Duration: | 4 s |
| Root Job Instance | 1567547 | Talend Root PID: | 20150622101208_4v0UB | | |
| Job Info: | AI_CORE/ai_smar_fetch_ads_image_url_one_customer-1.2:Prod | | | | |
| Return Code: | 4 | | | | |
| Job Server: | talend38-8 | | | OS PID: | 28558 |
| OS User: | tisadmin | | | | |
| Work Item: | 703114#15627 | | | | |
| Time Range Start: | | Time Range End: | | | |
| Value Range Start: | | Value Range End: | | | |
| Job Result: | | | | | |
| Count Input: | 0 | Count Output: | 1 | Count Updated: | 0 |
| Count Rejects: | 0 | Count Deleted: | 0 | | |
| Return Message: | tDie_1:Job failed. Last ad-id: 208578875<br>tMysqlInput_1:Communications link failure | | | | |
| View Log Table | The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server. | | | | |

# Agenda

1. Different Kind of Data Integration Jobs
2. Requirements
3. Overview about Components and Data Model
4. Using Components for Simple Jobs
5. Using Components for Incremental Loads
6. Using Components for Partinioned Data Loads
7. Monitoring Reports
8. Logging
9. Outlook

# Logging

- Every job has its own logger
- Every job instance has its own appender
  - Allows one file per job run
  - Allows distinct path per environment
  - Layout pattern can be vary from job to job if necessary
- Appender are flushed and closed at the end of the job instance
  - Appender to database (JOB_INSTANCE_LOGS table)
  - Appender to files
    - Pattern layout can contain context variables
    - Log file path can be set based on context variables
- Custom log4j.xml configuration can be used
- See the component documentation for configuration details

# Agenda

| | |
|---|---|
| 1 | Different Kind of Data Integration Jobs |
| 2 | Requirements |
| 3 | Overview about Components and Data Model |
| 4 | Using Components for Simple Jobs |
| 5 | Using Components for Incremental Loads |
| 6 | Using Components for Partinioned Data Loads |
| 7 | Monitoring Reports |
| 8 | Logging |
| 9 | Outlook |

# Outlook

- Additional table for work items within a job
  - Solves the problem for very frequent jobs processing a high number of work items with the fast growing values space of job_instance_ids
- Web Service to concentrate the database actions.
  - Solves the problem of huge amount parallel database sessions and allows caching of pre selected job instance ids
- Reporting will be enhanced with gant diagrams to display the scheduling
- Dump of context variables (without passwords)
- Make the Job Logger available for code within the job
- Use alternatively the logger from the job in case Talend has created one of instead of create a new one
- Watching the Log4j configuration file for changes at runtime
- Access the Log4j configuration via JMX
- Provide runtime information via JMX
- Migration to Log4j Version 2